Wrangling Data from OpenStreetMap with MongoDB

Map Area: San Francisco, CA, United States

1. Problems Encountered in the Map

The problem that I found while auditing the OpenStreetMap data of San Francisco was that there were many street names which included various forms abbreviations for a single word. For example, the word ‘Avenue’ was found to be abbreviated in two different forms: “Ave” and “Ave.”. There were also some misspellings for ‘Avenue’, for e.g. ‘Abenue’ and ‘Avenie’. This could be a problem when we want to find all the street names that are of type ‘avenue’ in some query later on as we would miss to take into consideration the streets with these abbreviated form, unless we correct them now before storing them into the database.

So I decided to clean them up by replacing many of the abbreviations used in street names with their full names. Here is a table of names that I used for correcting and expanding the abbreviated street names.

Abbreviated form or misspells	**Corrected Full Name**
Dr	Drive
ave., ave, avenie, abenue	Avenue
st., st	Street
Blvd., blvd, blvd,	Boulevard
Ct	Court
Rd.	Road

Here is an excerpt of the code that I used to correct the erroneous and abbreviated words in street names.

```
mapping = { "Dr": "Drive",
            "Ave": "Avenue",
            "Avenie": "Avenue",
            "Abenue": "Avenue",
            "St": "Street",
            "Blvd": "Boulevard",
            "CT": "Court",
            "Rd": "Road"
            }
```

```
def update_name(name, mapping):
    for (bad,correction) in mapping.items():
        bad = bad.lower()
        nameInLowerCase = name.lower()
        if nameInLowerCase.endswith(bad) or nameInLowerCase.endswith(bad+
".") or nameInLowerCase.endswith(bad + ","):
            name = street_type_re.sub(correction,name)
    return name
```

Another problem which I found in the dataset was that there were many nodes in it which were marked with postal codes which doesn't belong to San Francisco. San Francisco has postal codes starting with 941. I believe this from the information presented on this page: http://www.zip-codes.com/city/CA-SAN-FRANCISCO.asp
But there are far more many elements present in this dataset whose postal code doesn't start with 941. This is evidenced by following queries.

```
# Number of elements with postal code
> db.sf.find({"address.postcode": {"$exists":1}}).count()
11822

# Number of elements with postalcode starting with 941
> db.sf.find({"address.postcode": {"$regex": /^941/}}).count()
4877

# Postal codes ordered by count, the highlighted postal codes that don't
belong to San Francisco as they don't start with 941
> db.sf.aggregate([{"$match": {"address.postcode": {"$exists":1}}},
{"$group": {"_id": "$address.postcode", "count": {"$sum":1}}},{"$sort":
{"count": -1}}])
{ "_id" : "94611", "count" : 2981 }
{ "_id" : "94116", "count" : 1839 }
{ "_id" : "94610", "count" : 1353 }
{ "_id" : "94127", "count" : 595 }
{ "_id" : "94103", "count" : 407 }
{ "_id" : "94109", "count" : 381 }
{ "_id" : "94063", "count" : 373 }
{ "_id" : "94587", "count" : 257 }
{ "_id" : "94122", "count" : 209 }
{ "_id" : "94114", "count" : 177 }
{ "_id" : "94061", "count" : 171 }
{ "_id" : "94110", "count" : 134 }
{ "_id" : "94123", "count" : 127 }
{ "_id" : "94102", "count" : 125 }
```

```
{ "_id" : "94113", "count" : 112 }
{ "_id" : "94612", "count" : 107 }
{ "_id" : "94107", "count" : 106 }
{ "_id" : "94501", "count" : 102 }
{ "_id" : "94108", "count" : 101 }
{ "_id" : "94117", "count" : 94 }
```

The possible reason for presence of these extraneous postal codes can be that the ready-to-export map data available on OpenStreetMap site are usually built on rectangular boundary determined by the coordinates of the four corners of the rectangular covering the area of San Francisco. Because no region is perfectly rectangular in shape, a sample taken from rectangle from OpenStreetMap dataset is bound include some extraneous elements from surrounding region. One can remove such elements from dataset, if required before performing any analysis on it.

I also performed an aggregation query to check for if there were some malformed postal codes present in the dataset and my suspicion was true.

```
# The least frequently appearing postal codes in the dataset, highlighted ones look malformed
> db.sf.aggregate([{"$match": {"address.postcode": {"$exists":1}}},
{"$group": {"_id": "$address.postcode", "count": {"$sum":1}}},{"$sort":
{"count": 1}}])
{ "_id" : "94130", "count" : 1 }
{ "_id" : "90214", "count" : 1 }
{ "_id" : "94118-1316", "count" : 1 }
{ "_id" : "CA 94607", "count" : 1 }
{ "_id" : "94121 ", "count" : 1 }
{ "_id" : "94115 ", "count" : 1 }
{ "_id" : "94117-9991", "count" : 1 }
{ "_id" : "94017", "count" : 1 }
{ "_id" : "94301-2019", "count" : 1 }
{ "_id" : "25426", "count" : 1 }
{ "_id" : "515", "count" : 1 }
{ "_id" : "94720-1076", "count" : 1 }
{ "_id" : "94549-5506", "count" : 1 }
{ "_id" : "94519", "count" : 1 }
{ "_id" : "CA:94103", "count" : 1 }
{ "_id" : "94005", "count" : 1 }
{ "_id" : "CA 94404", "count" : 1 }
{ "_id" : "CA 94603", "count" : 1 }
{ "_id" : "944023025", "count" : 1 }
{ "_id" : "CA 94111", "count" : 1 }
```

Many of these malformed postal codes can be corrected by removing “CA “ and “CA:” prefix from them.

```
# Check how many postal codes has prefix CA
> db.sf.find({"address.postcode": {"$regex": /^CA/}}).count()
16
```

Below is the code excerpt that I included in project to correct the postcodes

```
# Corrects postal code by removing prefixes and suffixes
def update_postcode(postcode):
    postcode = re.sub("-\d+",'',postcode) #removes "-3131" from
"94121-3131"
    postcode = re.sub("^[^\d]+", "", postcode) #remove "CA " from "CA
94605"
    return postcode
```

I imported this corrected dataset in MongoDB.

2. Data Overview

Here is the statistics of data that I retrieved from OpenStreetMap.
Area: San Francisco, CA, United States

File Size:

san-francisco_california.osm 654 MB
san-francisco_california.osm.json 747 MB

Name of Collection in MongoDB: “sf”

Number of unique users:

```
> db.runCommand ( { distinct: "sf", key: "created.user" } ).values.length
1986
```

Number of documents:

```
> db.sf.find().count()
3367084
```

Number of nodes:

```
> db.sf.find({"type":"node"}).count()
3039345
```

Number of ways:

```
> db.sf.find({"type":"way"}).count()
327621
```

Number of company offices:

This turned out to be a much lower number than my expectation.

```
> db.sf.find({"office": {"$in": ["company"]}}).count()
74
```

Number of hiking trails:

```
> db.sf.find({"sac_scale": {"$in": ["hiking", "mountain_hiking"
,"demanding_mountain_hiking", "alpine_hiking", "demanding_alpine_hiking",
"difficult_alpine_hiking"]}}).count()
66
```

3. Additional Ideas

- There are publicly available datasets like Google's Freebase, which contain detailed information about many of the named locations on this planet. If we link this dataset with Freebase by including the unique ID of each city and location from Freebase with its corresponding element in OpenStreetMap database, then it can significantly improve the usability of this dataset by allowing to easily retrieve additional information about any location from either of the databases.
    - For example this dataset doesn't include information about what languages are spoken in the region, but this information is available in Freebase. One can retrieve this information for given region if the data is linked with Freebase by adding reference IDs to corresponding entities in Freebase as attributes to elements in OpenStreetMap dataset.
    - This linking process can be done programmatically while taking care of potential inaccuracy by making sure that longitude and latitude of corresponding locations roughly match in both the databases before linking them.
- I observed in the dataset that very few company offices are marked. In the dataset of San Francisco, I found only 74 company offices marked. This is quite contrary to the fact that San Francisco is a technology hub, so it must be containing a large number of offices for tech companies. Contributors can add this missing information in the dataset to improve its usability.